Exam File Provided By
The UofS IEEE Student Branch
ieee.usask.ca

EE431 Mi

Date: Tuesday, November 7, 2000
Time = 1:30 minutes
Text books and notes allowed.
Floppy Disks and hard disk verilog files allowed.

IMPORTANT: If an email is sent during the exam the sender will be disqualified and given a grade of zero. All email transactions will be monitored by academic computing services and will be checked after the exam.

Instructions

1. Modify your computational unit (which here means the instruction register and decoder circuit as well) in two ways.
   (a) Make the data bus (i.e. the output of the data bus mux) an output port.
   (b) For ALU function code 3'b000, make the operation r = x + y and for ALU function code 3'b001 make the operation r = x - y.
2. Copy the verilog file comp_unit_tester.v from G:classes\EE431 to the computational unit project folder on your own drive.
3. Modify comp_unit_tester.v as instructed below. The Verilog HDL comp_unit_tester.v instantiates the computational unit (It is the last statement in the module). The instantiation statement (for a particular computational unit) is given below:

```
comp_unit_int_ram c_u_1 (
                    .clk(clk),
                    .data_bus(data_bus),
                    .zero_flag(zero_flag),
                    .jump_instr(jump_inst),
                    .jump_not_z_inst(jump_not_zero_inst),
                    .PM_data(PM_data)
                     );
```

Replace this statement with a statement that instantiates your computational unit.

Notice that the address for the jump instructions (lower 4 bits of the IR register), which would be used by the program sequencer, is not used by the test program and is not in the connection list of the instantiation above.

4. Compile comp_unit_tester.v for a FLEX10K FPGA. It does not matter which one. You can have the compiler auto select the specific part.

5. Open the waveform editor and enter the nodes in the port list (others if you wish) of module comp_unit_tester. The module port list and port declarations for comp_unit_tester are given below:

```
module comp_unit_tester(
                    clk,
                    data_bus,
                    zero_flag,
                    jump_inst,
                    jump_not_zero_inst,
                    instr_cntr,
                    key);
input clk;
input [3:0] key;

output zero_flag, jump_inst, jump_not_zero_inst;
output [4:0] data_bus;
output [7:0] instr_cntr;
```

The output "instr_cntr" is part of the test program. It counts the instructions as they are passed to the instruction register. It is incremented with the positive edge of the clock. It is used to specify the time of interest in the questions.

6. While the waveform editor is active, set the end time of your simulation to 200 microseconds.

7. Using the waveform editor specify signal "clk" as a clock with period one microsecond that starts low.

8. Set input "key" to a constant. Use the value given in the table below.

9. Run the simulation and fill out the blank entries in the table. Two columns have been filled in as an example of what is required. You are asked the for the values of the data bus, zero flag and two jump instructions over the interval that "instr_cntr" is as specified. Please use Hexadecimal notation for the data bus values.

key (hex)	1 H								
instr_cntr (hex)	06H	09H	11H	49H	89H	96H	8EH	C0H	C1H
data_bus (hex)		—		8H					
zero_flag (bin)		0		1					
jump_inst (bin)		1		0					
jump_not_zero_inst (bin)		0		0					

7. Using the waveform editor specify signal "clk" as a clock with period one microsecond that starts low.

8. Set input "key" to a constant. Use the value given in the table below.

9. Run the simulation and fill out the blank entries in the table. Two columns have been filled in as an example of what is required. You are asked the for the values of the data bus, zero flag and two jump instructions over the interval that "instr_cntr" is as specified. Please use Hexadecimal notation for the data bus values.

key (hex)	2H								
instr_cntr (hex)	06H	09H	11H	49H	89H	A9H	BEH	C0H	C1H
data_bus (hex)		–		8H					
zero_flag (bin)		0		1					
jump_inst (bin)		1		0					
jump_not_zero_inst (bin)		0		0					

7. Using the waveform editor specify signal "clk" as a clock with period one microsecond that starts low.

8. Set input "key" to a constant. Use the value given in the table below.

9. Run the simulation and fill out the blank entries in the table. Two columns have been filled in as an example of what is required. You are asked the for the values of the data bus, zero flag and two jump instructions over the interval that "instr_cntr" is as specified. Please use Hexadecimal notation for the data bus values.

key (hex)	5H.								
instr_cntr (hex)	06H	09H	11H	49H	56H	89H	BEH	C0H	C1H
data_bus (hex)		–		8H					
zero_flag (bin)		0		1					
jump_inst (bin)		1		0					
jump_not_zero_inst (bin)		0		0					

7. Using the waveform editor specify signal "clk" as a clock with period one microsecond that starts low.

8. Set input "key" to a constant. Use the value given in the table below.

9. Run the simulation and fill out the blank entries in the table. Two columns have been filled in as an example of what is required. You are asked the for the values of the data bus, zero flag and two jump instructions over the interval that "instr_cntr" is as specified. Please use Hexadecimal notation for the data bus values.

key (hex)	6H								
instr_cntr (hex)	06H	09H	11H	49H	69H	89H	BEH	C0H	C1H
data_bus (hex)		—		8H					
zero_flag (bin)		0		1					
jump_inst (bin)		1		0					
jump_not_zero_inst (bin)		0		0					

7. Using the waveform editor specify signal "clk" as a clock with period one microsecond that starts low.

8. Set input "key" to a constant. Use the value given in the table below.

9. Run the simulation and fill out the blank entries in the table. Two columns have been filled in as an example of what is required. You are asked the for the values of the data bus, zero flag and two jump instructions over the interval that "instr_cntr" is as specified. Please use Hexadecimal notation for the data bus values.

key (hex)	FH								
instr_cntr (hex)	06H	09H	11H	49H	89H	9EH	BEH	C0H	C1H
data_bus (hex)		—		8H					
zero_flag (bin)		0		1					
jump_inst (bin)		1		0					
jump_not_zero_inst (bin)		0		0					

key (hex)	1								
instr_cntr (hex)	06	09	11H	49H	89	96H	BE	C0	C1
data_bus (hex)	1	—	1	8	—	0	4	E	7
zero_flag (bin)	0	0	0	1	0	1	0	0	0
jump_inst (bin)	0	1	0	0	1	0	0	0	0
jump_not_zero_inst (bin)	0	0	0	0	0	0	0	0	0

key (hex)	2								
instr_cntr (hex)	06	09	11	49	89	A9	BE	C0	C1
data_bus (hex)	1	—	1	8	—	4	5	C	7
zero_flag (bin)	0	0	0	1	0	1	0	0	0
jump_inst (bin)	0	1	0	0	1	0	0	0	0
jump_not_zero_inst (bin)	0	0	0	0	0	0	0	0	0

key (hex)	5								
instr_cntr (hex)	06	09	11	49	56	89	BE	C0	C1
data_bus (hex)	1	—	1	8	0	—	8	6	7
zero_flag (bin)	0	0	0	1	1	0	0	0	0
jump_inst (bin)	0	1	0	0	0	1	0	0	0
jump_not_zero_inst (bin)	0	0	0	0	0	0	0	0	0

key (hex)	6								
instr_cntr (hex)	06	9	11	49	69	89	BE	C0	C1
data_bus (hex)	1	—	1	8	C	—	9	4	7
zero_flag (bin)	0	0	0	1	1	0	0	0	0
jump_inst (bin)	0	1	0	0	0	1	0	0	0
jump_not_zero_inst (bin)	0	0	0	0	0	0	0	0	0